Aprendizaje de conocimiento rebatible

Telma Delladio

Email: td@cs.uns.edu.ar

*Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA)**

Dpto. de Ciencias e Ingeniería de la Computación (DCIC)

Universidad Nacional de Sur (UNS)

Avda. Alem 1254 - Bahía Blanca - Buenos Aires

Tel: (0291) 459-5135 - Fax: (0291) 459-5136

Workshop de Agentes y Sistemas Inteligentes

Palabras clave: Aprendizaje a partir de ejemplos - Programación en Lógica Rebatible Programación en Lógica Inductiva

Resumen

El aprendizaje de conceptos a partir de ejemplos es una tarea muy compleja. Dentro del área de la *Programación en Lógica Inductiva* se han desarrollado diversas propuestas para afrontar tal problema utilizando paradigmas tradicionales de programación en lógica. En este trabajo se propone la utilización de Programación en Lógica Rebatible como herramienta de representación de conocimiento y razonamiento no monótono a partir de la cual desarrollar aprendizaje inductivo de conceptos. La *Programación en Lógica Rebatible* como herramienta de razonamiento no monótono, ofrece una mayor flexibilidad al momento de sintetizar definiciones que modelen los conceptos que son objeto de aprendizaje.

1 Programación en Lógica Rebatible

La Programación en Lógica Rebatible (PLR) combina programación en lógica con argumentación rebatible. La PLR permite la representación de conocimiento a través de reglas estrictas y reglas rebatibles. Las *reglas rebatibles*, representan información débil o tentativa. Una regla rebatible de la forma $A \prec B.$ es utilizada en la práctica para representar conocimiento rebatible. Esto es, conocimiento que puede considerarse válido solo si no hay razones fundadas para considerar

* Miembro del IICyTI (Instituto de Investigación en Ciencia y Tecnología Informática)

lo contrario. Por ejemplo, $\neg$ *vuela* $\prec$ *mamífero.*. Las *reglas estrictas*, por otro lado, son de la forma $X \leftarrow Y$ y representan información no rebatible, como por ejemplo *cetáceo* $\leftarrow$ *delfín*.

Un programa lógico rebatible (*p.l.r*) P es un par (Π, Δ), donde Π es el conjunto de información no rebatible representada por reglas estrictas y hechos lógicos, mientras que Δ representa el conocimiento rebatible a través de un conjunto de reglas rebatibles. La PLR ofrece un procedimiento de justificación que implementa un análisis dialéctico evaluando argumentos a favor y en contra de aquellas piezas de conocimiento bajo discusión. De esta forma un literal L se dice que estará *garantizado* si existe un argumento para L que no puede ser derrotado en forma efectiva. En la PLR un argumento $\mathcal{A}$ para un literal p, denotado $\langle \mathcal{A}, p \rangle$, es un conjunto minimal de reglas rebatibles consistente con el conocimiento estricto disponible (reglas estrictas y hechos) y que junto a este permite obtener una derivación para p. En este caso se dice que el argumento $\mathcal{A}$ *soporta* al literal p. Cualquier argumento formado únicamente por reglas rebatibles utilizadas en un argumento $\mathcal{A}$ se denomina subargumento de $\mathcal{A}$.

El análisis diálectico utilizado en la PLR considera todos los posibles argumentos en contra del argumento $\mathcal{A}$ que soporta p. Si existe algún argumento $\mathcal{B}$ que está en contra del argumento $\mathcal{A}$, y es lo suficientemente bueno, se dice que el argumento $\mathcal{B}$ derrota al argumento $\mathcal{A}$. Pero a su vez, si este argumento $\mathcal{B}$ es derrotado por un tercer argumento, el argumento $\mathcal{A}$ prevalecerá. El análisis dialéctico considerará entonces todos los derrotadores de $\mathcal{A}$ y luego los derrotadores de estos últimos y continuará con este procedimiento hasta determinar efectivamente si $\mathcal{A}$ prevalece o no. Para realizar este análisis se define la noción de *ataque* entre argumentos. Se dice que un argumento $\langle \mathcal{B}, q \rangle$ ataca a un argumento $\langle \mathcal{A}, p \rangle$, si existe un subargumento de $\mathcal{A}$, $\langle \mathcal{C}, r \rangle$, y se sabe que $\{q, r\}$ es inconsistente con el conocimiento estricto disponible. Dicho de otra forma, $\mathcal{B}$ soporta un literal (q) que va en contra de un literal (r) utilizado en la formación del argumento $\mathcal{A}$. La noción de ataque no es suficiente para determinar una relación de derrota entre argumentos. Es necesario tener un criterio de preferencia entre argumentos. Con estas dos nociones, ataque y preferencia entre argumentos se define la noción de *derrota*. Entonces, si existe un argumento $\mathcal{A}$ que es atacado en alguno de sus subargumentos $\mathcal{A}_{sub}$ por un argumento $\mathcal{B}$ y de acuerdo al criterio utilizado $\mathcal{B}$ es preferido a $\mathcal{A}_{sub}$, se dice que $\mathcal{B}$ es un *derrotador propio* de $\mathcal{A}$. En caso que ninguno de los argumentos considerados, $\mathcal{A}_{sub}$ y $\mathcal{B}$, sea preferido por sobre el otro, se dice que $\mathcal{B}$ *bloquea* a $\mathcal{A}$ y es un derrotador por bloqueo. En la PLR se utiliza *especificidad* [GS03] como criterio de comparación entre argumentos. Intuitivamente, este criterio favorece a aquellos argumentos que utilizan mayor información del dominio y utilizan menos reglas. Dicho de otra forma, se prefieren aquellos argumentos que son más precisos y concisos.

2 Aprendizaje y PLR

Dentro del área de la *Programación en Lógica Inductiva* (ILP) se han desarrollado diversas técnicas y métodos de síntesis de programas lógicos que sirvan como definición de uno o más

conceptos. Esta síntesis es guiada por un conjunto de observaciones que son ejemplos de instancias del concepto que se desea aprender. En general, cuando se desarrollan tareas de aprendizaje a partir de ejemplos suelen utilizarse dos clases de ejemplos. Por un lado se utilizan ejemplos u observaciones positivas y por otro lado se utilizan ejemplos negativos. Las observaciones clasificadas como positivas son instancias ciertas del concepto que se desea aprender mientras que aquellas observaciones clasificadas como negativas describen instancias que no pertenecen a ese concepto. Las observaciones positivas sirven para obtener información acerca de las propiedades que cumplen los objetos pertenecientes al concepto mientras que las observaciones negativas sirven para identificar propiedades que no debieran cumplirse en instancias del concepto. Tradicionalmente las observaciones se dividen en dos conjuntos disjuntos, un conjunto de observaciones positivas y un conjunto de observaciones negativas. Por ejemplo, los siguientes conjuntos de observaciones positivas (O^+) y negativas (O^-) podrían ser utilizados en el problema de aprender el concepto "*par*":

$$O^+ = \{\ \text{par}(2).,\ \text{par}(4).,\ \text{par}(8).\ \}$$
$$O^- = \{\ \text{par}(1).,\ \text{par}(7).,\ \text{par}(5).\ \}$$

En el contexto de la PLR, no consideraremos explícitamente la separación entre observaciones positivas y negativas. Al contar con la posibilidad de representar información negativa explícitamente a través de literales negativos, esta distinción, en principio, no es necesaria. De esta forma, para ese mismo problema se puede considerar el siguiente conjunto unificado de observaciones:

$$O = \{\ \neg\ \text{par}(1).,\ \text{par}(2).,\ \text{par}(4).,\ \neg\ \text{par}(5).,\ \neg\ \text{par}(7).,\ \text{par}(8).\ \}.$$

El objetivo, como en toda propuesta de aprendizaje a partir de ejemplos, sigue siendo el mismo. Encontrar una definición adecuada, en el lenguaje de representación utilizado, que modele el concepto descripto por un conjunto de observaciones.

2.1 El problema del aprendizaje en el marco de la PLR

El problema del aprendizaje a partir de ejemplos puede plantearse en el marco de la PLR. En este escenario, todo concepto C que pueda ser objeto de aprendizaje, es convenientemente modelado por reglas rebatibles cuya cabeza será, un literal positivo $c(x_1, \ldots, x_n)$ o un literal negativo $\neg\ c(x_1, \ldots, x_n)$. En cada caso c es un átomo predicativo n_ario del lenguaje utilizado para representar el concepto real C.

Consideraremos una definición del problema de aprendizaje a través de ejemplos en el marco de la PLR. En este caso el problema de aprendizaje puede definirse de la siguiente manera:

Definición 2.1: Problema de aprendizaje en el marco PLR
Dados:

- un programa lógico rebatible $\mathcal{BC} = (\Pi, \Delta)$
- un conjunto de observaciones $\mathcal{O}$
- un espacio de hipótesis $\mathcal{H}$

Encontrar:

- un conjunto de reglas rebatibles $S \in \mathcal{H}$, tal que $\forall o \in \mathcal{O}$, se verifique que el *p.l.r* $(\Pi, \Delta \cup S)$ cubra o

□

Una instancia particular del problema de aprendizaje de conceptos en el marco PLR lo notaremos con una tupla $\langle \Pi, \Delta, \mathcal{H}, \mathcal{O} \rangle$. El programa lógico rebatible $\mathcal{BC}$ representa el conocimiento previo disponible, ya sea conocimiento estricto (Π) o rebatible (Δ). El conjunto de observaciones $\mathcal{O}$ es un conjunto de hechos (información no rebatible) que hacen referencia únicamente al concepto que se desea aprender. Asumimos que $\mathcal{O} \cup \Pi$ es no contradictorio. El espacio de hipótesis $\mathcal{H}$ determina qué reglas rebatibles pueden utilizarse para conformar la definición del concepto que se desea aprender, de esta forma se restringe el tipo de definiciones que pueden considerarse. Por esa razón el resultado del aprendizaje, el conjunto de reglas rebatibles S, será un elemento de este espacio de hipótesis. La condición " $(\Pi, \Delta \cup S)$ cubra o" quiere significar que la definición hallada deber servir, junto con el conocimiento previo, para sancionar cada una de las observaciones consideradas. Dicho de otra forma, cada una de las observaciones debe ser una consecuencia, una conclusión válida, del programa lógico rebatible $(\Pi, \Delta \cup S)$.

Como el formalismo de representación de conocimiento y razonamiento utilizado es la PLR consideraremos que un *p.l.r* cubre un literal si este literal está *garantizado* [GS03] por dicho programa.

Ejemplo 2.1:
Considere la siguiente instancia del problema de aprendizaje, $\langle \Pi_1, \Delta_1, \mathcal{H}_1, \mathcal{O}_1 \rangle$.

- $\mathcal{BC}_1 = (\Pi_1, \Delta_1)$ representa el conocimiento previo, donde:
  $\Pi_1 = \{$gato(`reimundo`)., perro(`topper`)., canario(`crispín`).$\}$
  $\Delta_1 = \{$mascota(X) $\prec$ perro(X)., mascota(X) $\prec$ gato(X).$\}$
- El espacio de hipótesis $\mathcal{H}_1$ está determinado por aquellas reglas rebatibles que pueden formarse utilizando los literales y términos en la base de conocimiento previo $\mathcal{BC}_1 = (\Pi_1, \Delta_1)$ y en el conjunto de observaciones $\mathcal{O}_1$.
- $\mathcal{O}_1 = \{$ mimoso(`topper`)., $\neg$ mimoso(`crispín`)., mimoso(`reimundo`). $\}$
  El conjunto de observaciones $\mathcal{O}_1$ indica que el concepto que se desea aprender es "*mimoso*".

Luego, una solución al problema $\langle \Pi_1, \Delta_1, \mathcal{H}_1, \mathcal{O}_1 \rangle$ es:

$$S_1 = \{\text{mimoso(X)} \prec \text{ mascota(X).}, \neg \text{ mimoso(X)} \prec \text{ canario(X).}\}$$

■

En el ejemplo anterior se puede observar que el conjunto obtenido, S_1, cumple con las condiciones buscadas pues todas las observaciones son cubiertas (garantizadas) por el programa lógico rebatible $(\Pi_2, \Delta_2 \cup S_1)$. Decimos que S_1 es una definición rebatible completa para el concepto "*mimoso*".

Definición 2.2: Definición rebatible para el concepto C
Llamaremos definición rebatible para un concepto C a cualquier conjunto de reglas rebatibles cuyas cabezas son de la forma $c(x_1, \ldots, x_n)$ o $\neg\, c(x_1, \ldots, x_n)$, donde c es el átomo predicativo del lenguaje utilizado para representar el concepto C. □

Definición 2.3: Completitud
Dado un problema de aprendizaje $\langle \Pi, \Delta, \mathcal{H}, \mathcal{O} \rangle$,

- decimos que la definición $C \in \mathcal{H}$ es incompleta si: $\exists o \in \mathcal{O}$ tal que o no está garantizado por el *p.l.r* $(\Pi, \Delta \cup C)$
- decimos que la definición $C \in \mathcal{H}$ es completa si: $\forall o \in \mathcal{O}$ se verifica que o está garantizado por el *p.l.r* $(\Pi, \Delta \cup C)$

□

Definición 2.4: Conflicto
Dado un problema de aprendizaje $\langle \Pi, \Delta, \mathcal{H}, \mathcal{O} \rangle$, decimos que existe un conflicto entre una definición $C \in \mathcal{H}$ y el conjunto de observaciones $\mathcal{O}$ si:

- o está garantizado por el *p.l.r* $(\Pi, \Delta \cup C)$ y $\bar{o} \in \mathcal{O}$,

Con $\bar{o}$ denotamos al complemento del literal o. □

Obsérvese que si una definición rebatible es conflictiva también es incompleta. Dicho de otra forma, si una definición rebatible es completa no es conflictiva.

Definición 2.5: Clasificación de observaciones
Dado un problema de aprendizaje $\langle \Pi, \Delta, \mathcal{H}, \mathcal{O} \rangle$ y una definición rebatible $C \in \mathcal{H}$, el conjunto de observaciones $\mathcal{O} = \mathcal{O}_f \cup \mathcal{O}_{ok} \cup \mathcal{O}_c$, donde:

- $\mathcal{O}_f = \{o \in \mathcal{O}$: $(\Pi, \Delta \cup C)$ no garantiza $o\ \}$
- $\mathcal{O}_{ok} = \{o \in \mathcal{O}$: $(\Pi, \Delta \cup C)$ garantiza $o\ \}$
- $\mathcal{O}_c = \{o \in \mathcal{O}$: $(\Pi, \Delta \cup C)$ garantiza $\bar{o}\ \}$

En la definición anterior, el conjunto $\mathcal{O}_f$ representa el conjunto de observaciones que no son cubiertas por la definición C. El conjunto $\mathcal{O}_{ok}$ representa las observaciones que son cubiertas correctamente. Por último, $\mathcal{O}_c$ es el conjunto de observaciones que presentan conflicto con la definición C, decimos que son observaciones cubiertas en forma errónea. Obsérvese que $\mathcal{O}_c \subseteq \mathcal{O}_f$.

El próximo ejemplo ilustra los términos introducidos en las definiciones anteriores.

Ejemplo 2.2: [1]
Considere la siguiente instancia del problema de aprendizaje, $\langle \Pi_2, \Delta_2, \mathcal{H}_2, \mathcal{O}_2 \rangle$.

- Π_2 = { paloma(`pio`)., paloma(`pepe`)., gallina(`tina`)., gallina(`turu`)., bebé(`pepe`)., pingüino(`tweety`)., asustada(`tina`). pájaro(X)← paloma(X)., pájaro(X)← gallina(X)., pájaro(X)← pingüino(X)., ¬ vuela(X) ← pingüino(X). }
- $\Delta_2 = \emptyset$
- $\mathcal{O}_2$ = { vuela(`tina`), vuela(`pio`), vuela(`turu`), ¬ vuela(`tweety`), ¬ vuela(`pepe`) }
- $\mathcal{H}_2$ definido por las reglas rebatibles que pueden formarse utilizando los literales y términos en $\mathcal{BC}_2$=(Π_2,Δ_2) y $\mathcal{O}_2$.

La definición C_1 = {vuela(X) $-\prec$ pájaro(X).}, genera un conflicto ya que el literal "vuela(`pepe`)" está garantizado por $(\Pi_2, \Delta_2 \cup C_1)$ pero "¬ vuela(`pepe`)" $\in \mathcal{O}_2$. Dicho de otra forma, "¬ vuela(`pepe`)" $\in \mathcal{O}_c$. Por otro lado, hay que observar que la definición C_1 permite obtener una derivación rebatible [GS03] para "vuela(`tweety`)", esto no genera conflicto alguno debido a que si bien existe tal derivación, el literal "vuela(`tweety`)" no está garantizado por $(\Pi_2, \Delta_2 \cup C_1)$.

De igual forma, la definición C_2 = {¬ vuela(X) $-\prec$ gallina(X).} genera un conflicto pues el literal "¬ vuela(`turu`)" está garantizado por $(\Pi_2, \Delta_2 \cup C_2)$ mientras "vuela(`turu`)" $\in \mathcal{O}_2$. Esto es, "vuela(`turu`)" $\in \mathcal{O}_c$

Por otro lado la definición C_3 es completa ($\mathcal{O}_2 = \mathcal{O}_{ok}$) y por lo tanto no genera ningún conflicto:
C_3 = { vuela(X) $-\prec$ pájaro(X)., ¬ vuela(X) $-\prec$ gallina(X)., vuela(X) $-\prec$ paloma(X)., ¬ vuela(X) $-\prec$ paloma(X), bebé(X)., vuela(X) $-\prec$ gallina(X), asustada(X). }.

■

La solución a una instancia del problema de aprendizaje consiste en hallar un conjunto de reglas rebatibles que permita junto con el conocimiento previo garantizar las observaciones. Existen diversas propuestas en el área de ILP [MR94, NCd97, Bra90] para afrontar el problema de generar una definición adecuada del concepto en cuestión. En el marco de la PLR también se pueden definir mecanismos para generar una definición. En la siguiente sección se analiza el

[1] Ejemplo adaptado de [GS03]

problema de la generación de definiciones rebatibles para afrontar el problema de aprendizaje en el marco PLR.

2.2 Aprendizaje *Top Down* de reglas rebatibles

Se propone que el aprendizaje (generación) de reglas rebatibles siga un esquema *top down*. Dada una instancia particular del problema de aprendizaje es necesario generar un conjunto de reglas rebatibles que sirva como definición del concepto que se desea aprender.

La tarea de aprendizaje comienza considerando una definición general. Esta definición estará formada por una única regla rebatible lo más general posible. A partir de la regla inicial que se haya seleccionado, se procede con un mecanismo de ajuste o refinamiento el cual finaliza cuando las observaciones suministradas son cubiertas correctamente por el conjunto de reglas generadas. El proceso acaba cuando se obtiene una definición completa. Si se presenta algún conflicto entre la definición actualmente considerada y el conjunto de observaciones, la definición actual debe ser refinada. El objetivo en cada paso de refinamiento es eliminar tantos conflictos como sea posible para conducir el proceso de aprendizaje hacia la obtención de una definición completa.

La naturaleza no monótona de la PLR permite que el conjunto de literales sancionados pueda variar fácilmente con la modificación del conjunto de reglas rebatibles. Por eso, el proceso de refinamiento se basa en la modificación del conjunto de reglas rebatibles que conforman la definición actual. Este refinamiento consistirá, en la mayoría de los casos, en el agregado de una o más reglas rebatibles a la definición actual. La nuevas reglas que pasan a formar parte de la definición se originan a partir de alguna de las reglas ya presentes en la definición actual. Los conflictos son los que determinan el paso de especialización a realizar. Se debe analizar para cada elemento de $\mathcal{O}_c$, qué argumentos permiten que los complementos de estas observaciones sean garantizados. De esta forma se pueden identificar cuáles son las reglas rebatibles, dentro de la definición actual, que permiten garantizar (cubrir) el complemento de una observación. A partir de estas reglas se generarán las nuevas reglas rebatibles que pasarán a formar parte de la definición.

Son diversas las modificaciones que pueden realizarse en cada paso de refinamiento. Estas modificaciones dependen de las observaciones y las reglas rebatibles que se consideren en cada momento. Se analizan a continuación diversas situaciones y heurísticas que pueden utilizarse al momento de seleccionar un refinamiento de la definición actual.

Heurística 1 : *Cubrimiento erróneo*
Si una de las reglas rebatibles pertenecientes a la definición actual permite garantizar únicamente observaciones en forma errónea se puede incluir a la definición la regla obtenida de negar su consecuente.

Por ejemplo, dado el problema $\langle \emptyset, \emptyset, \mathcal{H}, \mathcal{O} \rangle$. Si $\mathcal{O}$={p(a).,p(b).} C = { $\neg$ p(X) $-\prec$.} es la definición actual, se genera la nueva definición C' = { p(X) $-\prec$., $\neg$ p(X) $-\prec$.}

Heurística 2 : *Bloqueo Mutuo*

Supongamos que considerando un problema $\langle \Pi, \Delta, \mathcal{H}, \mathcal{O} \rangle$ y una definición actual C, existen dos argumentos $\mathcal{A}$ y $\mathcal{B}$ incluidos en $\Delta \cup C$ que se bloquean mutuamente y que soportan o y $\bar{o}$ respectivamente. Se sabe además que $o \in \mathcal{O}$. En este caso, se puede eliminar la regla principal del argumento $\mathcal{B}$ para permitir que el argumento $\mathcal{A}$ que soporta o se convierta en un argumento que garantice o (esto únicamente si no existen otros derrotadores de $\mathcal{A}$). Obsérvese que si $\mathcal{B}$ era garantía de alguna observación, esta garantía ya no se mantiene con la eliminación de $\mathcal{B}$.

Por ejemplo, dado el problema $\langle \Pi, \emptyset, \mathcal{H}, \mathcal{O} \rangle$. Si Π = { r(a), r(b), s(a) }, el conjunto de observaciones es $\mathcal{O}$ = { p(a)., $\neg$ p(b) } y C = { $\neg$ p(X) $-\prec$ r(X)., p(X) $-\prec$ s(X). } es la definición actual, el argumento $\mathcal{A}$ = { p(X) $-\prec$ s(X).} soporta el literal "p(a)" mientras que el argumento $\mathcal{B}$ = { $\neg$ p(X) $-\prec$ r(X). } soporta el literal "$\neg$ p(a)". No obstante, ninguno de estos dos literales está garantizado. Eliminando de la definición actual la regla que conforma el argumento $\mathcal{B}$ se puede obtener la definición C'= { p(X) $-\prec$ s(X). } que garantiza el literal "p(a)", pero hay que notar que se pierde la garantía sobre el literal "$\neg$ p(b)" que aportaba C.

Heurística 3 : *Especialización de contra-argumentos*

Supongamos que la definición actual permite la formación de algún argumento que garantice un literal p con $\bar{p} \in \mathcal{O}_c$. Supongamos también que $\mathcal{A}$ es el argumento más específico que garantiza p, y $\mathcal{B}$ es el único argumento que garantiza una observación $q \in \mathcal{O}$, verificándose además que $A -\prec B_1, \ldots, B_n.$ es la regla principal en $\mathcal{A}$ y $\mathcal{B}$. En este caso no sería conveniente eliminar esa regla de la definición actual. Sería conveniente solucionar el conflicto causado por la observación $\bar{p}$ sin perder la garantía sobre q. Para ello se puede generar una regla de la forma $\neg A -\prec B_1, \ldots, B_n, E.$ que sirva en la formación de un argumento para $\bar{p}$. La especialización vía el literal E debe ser guiada por la observación $\bar{p}$ para no generar un argumento que derrote al argumento que utiliza $A -\prec B_1, \ldots, B_n.$ para soportar q puesto que esta última modificación generará posiblemente un argumento más específico que el original.

Por ejemplo, dado el problema $\langle \Pi, \emptyset, \mathcal{H}, \mathcal{O} \rangle$. Si Π = { g(a), g(b), s(a)}, $\mathcal{O}$= { v(a), $\neg$ v(b)} y la definición actual es C = { $\neg$ v(X) $-\prec$ g(X). } aplicando esta heurística puede obtenerse una nueva definición C' = { $\neg$ v(X) $-\prec$ g(X)., v(X) $-\prec$ g(X), s(X). } que garantiza ambas observaciones.

Heurística 4 : *Unfolding*

Supongamos que existe una regla rebatible en la definición actual que permite formar argumentos que garantizan observaciones correctas y erróneas. La aplicación de *unfolding* permite separar el conocimiento utilizado para soportar cada tipo de observaciones.

Por ejemplo, dado el problema $\langle \Pi, \emptyset, \mathcal{H}, \mathcal{O} \rangle$. Si Π = {p(X) $\leftarrow$ g(X), p(X) $\leftarrow$ t(X), g(a), g(b),

t(c)}, $\mathcal{O}$= { ¬ v(a), ¬ v(b), v(c) } y la definición actual es C = { v(X) $\prec$ p(X). }. La única regla de la definición permite cubrir algunas observaciones en forma correcta y otras en forma incorrecta. Por aplicación de *unfolding* se puede obtener la definición C' = { v(X) $\prec$ p(X)., v(X) $\prec$ t(X)., v(X) $\prec$ g(X). }. De esta forma se obtienen nuevas reglas que soportan solo observaciones en forma correcta o solo observaciones en forma errónea, facilitando posteriores pasos de refinamiento.

En la siguiente sección se analiza una instancia del problema de aprendizaje en el marco de la PLR y se muestra cómo pueden utilizarse algunas de estas heurísticas.

3 Un ejemplo: *buy stock*

El siguiente ejemplo [2] ilustra cómo puede llevarse a cabo la tarea de aprendizaje en el marco de la PLR. Considérese el problema de aprendizaje $\langle \Pi, \Delta, \mathcal{H}, \mathcal{O} \rangle$, donde:

- Π = { strong(steel)., closing(faber)., in fusion(tnt, faber)., in fusion(acme, steel)., good price (acme)., good price (tnt)., ¬ good price (faber)., ¬ good price (red)., ¬ good price (green). }
- Δ ={ risky company(C) $\prec$ in fusion(C,D)., risky company(C) $\prec$ closing(C)., ¬ risky company(C) $\prec$ in fusion(C,D), strong(D). }
- $\mathcal{H}$ determinado por las reglas rebatibles que pueden formarse utilizando los literales y términos en $\mathcal{BC}$=(Π, Δ) y $\mathcal{O}$.
- $\mathcal{O}$= { buy stock(acme)., ¬ buy stock(faber)., ¬ buy stock(tnt). }

De acuerdo al conjunto de observaciones, el concepto que se desea aprender es "*buy stock*".

Ya que el mecanismo de aprendizaje propuesto sigue una línea *top down*, se propone como primera definición rebatible aquella conformada por una única regla rebatible que sea lo suficientemente general. Por esto, la primera definición considerada es C_1={buy stock(X) $\prec$.}. Cabe destacar, que como las definiciones consideradas son conjuntos de reglas rebatibles, en el caso de C_1, la regla que lo conforma es una regla rebatible sin cuerpo. Este tipo de reglas rebatibles son conocidas como "*presunciones*" [GS03].

La definición C_1= {buy stock(X) $\prec$.} particiona el conjunto de observaciones de la siguiente forma:

- $\mathcal{O}_f$ = { ¬ buy stock (faber)., ¬ buy stock (tnt). }
- $\mathcal{O}_{ok}$ = { buy stock (acme). }
- $\mathcal{O}_c$ = {¬ buy stock (faber)., ¬ buy stock(tnt). }

[2] Ejemplo adaptado de [GS03]

Es necesario refinar C_1 con el objetivo de eliminar los conflictos existentes. Se puede aplicar la heurística 3 y para ello se necesita una propiedad (literal) que permita refinar el cuerpo de la regla que forma la definición actual. Informalmente decimos que una propiedad es simple si representa un atributo y se define en forma explícita, por ejemplo "*good price*". Por otro lado, consideramos que una propiedad es derivada, si se define en término de otras propiedades o atributos, por ejemplo "*risky*".

Un simple análisis muestra que no pueden utilizarse propiedades simples para eliminar los conflictos presentes. Por eso se utiliza la propiedad "*risky*". Esta propiedad permite separar las observaciones cubiertas correctamente, $\mathcal{O}_{ok}$, de aquellas que están siendo cubiertas en forma errónea, $\mathcal{O}_c$. Pues "risky company(X)" se verifica si X es `faber` o si X es `tnt`. Por el contrario, no se verifica si X es `acme`. Entonces, utilizando la regla "buy stock(X) $\prec$.", y la propiedad "*risky*", se genera la nueva regla "¬ buy stock(X) $\prec$ risky company(X).". Así se obtiene una definición completa:

$$C_2 = \{\text{buy stock(X)} \prec \text{., } \neg \text{ buy stock(X)} \prec \text{ risky(X).}\}.$$

verificando $\mathcal{O}_{ok} = \mathcal{O}$ y $\mathcal{O}_f = \mathcal{O}_c = \emptyset$.

Obsérvese que si se hubiese elegido como primera definición el conjunto { ¬ buy stock(X) $\prec$.}, siguiendo un proceso similar al anterior se hubiera obtenido una solución diferente al mismo problema:

$$\{\neg \text{ buy stock(X)} \prec \text{ ., buy stock(X)} \prec \neg \text{ risky company(X).}\}.$$

Consideremos ahora una nueva instancia del problema de aprendizaje, $\langle \Pi, \Delta \cup C_2, \mathcal{H}, \mathcal{O}' \rangle$, donde se consideran nuevas observaciones; $\mathcal{O}'$ = { ¬ buy stock(`red`)., buy(`green`)}.

Debido a que el conocimiento actual, $\mathcal{BC}$' = $(\Pi, \Delta \cup C_2)$, tiene un conjunto de reglas que sirve de definición para el concepto "*buy stock*", se analiza primero si existe conflicto entre las nuevas observaciones y la definición actual.

El literal "buy stock(`red`)" está garantizado por $\mathcal{BC}$' por lo tanto es necesario modificar la definición del concepto "*buy stock*". El conjunto de reglas que sirve de definición actual es C_2, y el único argumento que garantiza "buy stock(`red`)" es el formado por la regla regla "¬ buy stock(X) $\prec$.". Es necesario entonces obtener una regla rebatible que soporte y garantice correctamente la observación eliminando el argumento que garantiza su complemento. La propiedad utilizada para esto es "*good price*", ya que esta propiedad permite solucionar el conflicto en forma completa. La nueva regla rebatible será "¬ buy stock(X) $\prec$ ¬ good price(X).". Así el conjunto $C_3 = C_2 \cup \{\neg$ buy stock(X) $\prec$ ¬ good price(X).} forma una definición completa para el concepto "*buy stock*".

4 Conclusiones

En este trabajo se presenta la utilización de Programación en Lógica Rebatible como herramienta de representación de conocimiento y razonamiento no monótono a partir de la cual desarrollar aprendizaje inductivo de conceptos. La naturaleza no monótona de la PLR permite que el conjunto de literales sancionados pueda variar fácilmente con la modificación del conjunto de reglas rebatibles. Esto, junto con la propuesta de representar definiciones, precisamente, a través de un conjunto de reglas rebatibles permite que la tarea de aprendizaje sea más flexible. La consideración de nuevas observaciones se acota al refinamiento de la definición actual. Los refinamietos se basan en pequeñas modificaciones al conjunto de reglas rebatibles que conforman la definición de concepto en cuestión. Estas modificaciones involucran, en cada paso, la eliminación o agregado de algunas reglas rebatibles originadas en el análisis de los conflictos presentes. Estas modificaciones provocan que el conjunto de literales sancionados modele a cada paso, en forma más precisa el concepto que se está aprendiendo.

Referencias

[Bra90] Ivan Bratko. *Prolog Programming for Artifitial Intelligent.* Addison Wesley, 2nd edition, 1990.

[GS03] Alejandro J. García and Guillermo R. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 2003.

[MR94] Stephen Muggleton and Luc De Raedt. Inductive logic programing: Theory and methods. *Journal of Logic Programming*, 19(20), 1994.

[NCd97] Shan-Hwei Nienhuys-Cheng and Ronald deWolf. *Foundations of Inductive Logic Programming.* Springer, 1997.